Amendments to the Specification:

Paragraph [0008]

[0008] The present invention provides an automated software test system which, in its various embodiments, overcomes at least the deficiencies in previous test tools that are discussed above. A system in accordance with the present invention comprises a test tool running on a development computer (or PC) that communicates via a transmission medium with an agent executing on a target device. The target device also executes **[[ ]]**an application to be tested. In general, testing is performed by using the test tool on the development computer to execute a test script, which causes the development computer to generate a variety of test input to be injected via the agent into the application under test on the target device. The test tool is configured to validate whether actual output contained on the target device matches expected output stored in the test tool. In performing these functions, the test tool (via the agent) is able to send events to, and get information from, individual GUI controls, which are displayed on the target device.

Paragraph [0035]

[0035] The logging manager 41 also enables a tester to view only those message types that he/she would like to see. To provide this capability, the test tool 21 includes a log viewer which allows a [[ ]]user to filter a log file based on one or more message types. Error and warning messages may include screen comparison/expected value information. A bitmap "snapshot" of a screen on the target device 14 can be written to a log file. This information can be viewed by a user while a test script 22 is running or after the test has completed to determine whether any "bugs" were encountered. The log viewer, accessed through a test tool interface, was designed to help a test developer view any log files created by them or the test system 10. The log viewer also allows the user to view specific errors or warnings based on an error or warning number. In one embodiment, a default system log can always be open to receive all messages the test developer creates as well as all system level logging information. The default system log is created during a test initialization phase. Preferably, to prevent loss of this information, this log cannot be deleted using any of the methods or functions associated with the logging manager 41.

Paragraph [0037]

[0037] The trap manager 49 is responsible for "trapping" and handling the occurrence of unexpected screens (or "trap events") that can appear on the target device 14 during execution of an automated test. The ability to handle unexpected screens, message boxes or other such events can be an important aspect of conducting the unattended execution of automated tests. When creating a test script 22 which is designed to dictate a series of events that occur during an automated test, a test developer typically expects [[ ]]events to occur in a certain order so that he or she can effectively test particular aspects of the AUT 37. For the most part, the test developer is able to control the sequence of events via the commands in the test script 22. However, it is not uncommon to have trap events such as unexpected screens pop up, which can adversely impact the automated test, even causing it to fail.